Learnt Compression for Visual Analytics on the Edge

Yao Wang
Dept. of Electrical and Computer Engineering and Biomedical Engineering
Tandon School of Engineering
New York University
https://wp.nyu.edu/videolab

Supported in part by NSF and Intel through its MLWiNS (Machine Learning for Wireless Networks) program

Outline

- Motivation: Edge-Assisted visual analytics for mobile devices
- Two camps of approaches
  - Compression->Decompression->Visual analytics
  - Split computing and compress intermediate features
- Learnt feature compression for split computing
- Learnt scalable feature compression
- Conclusion

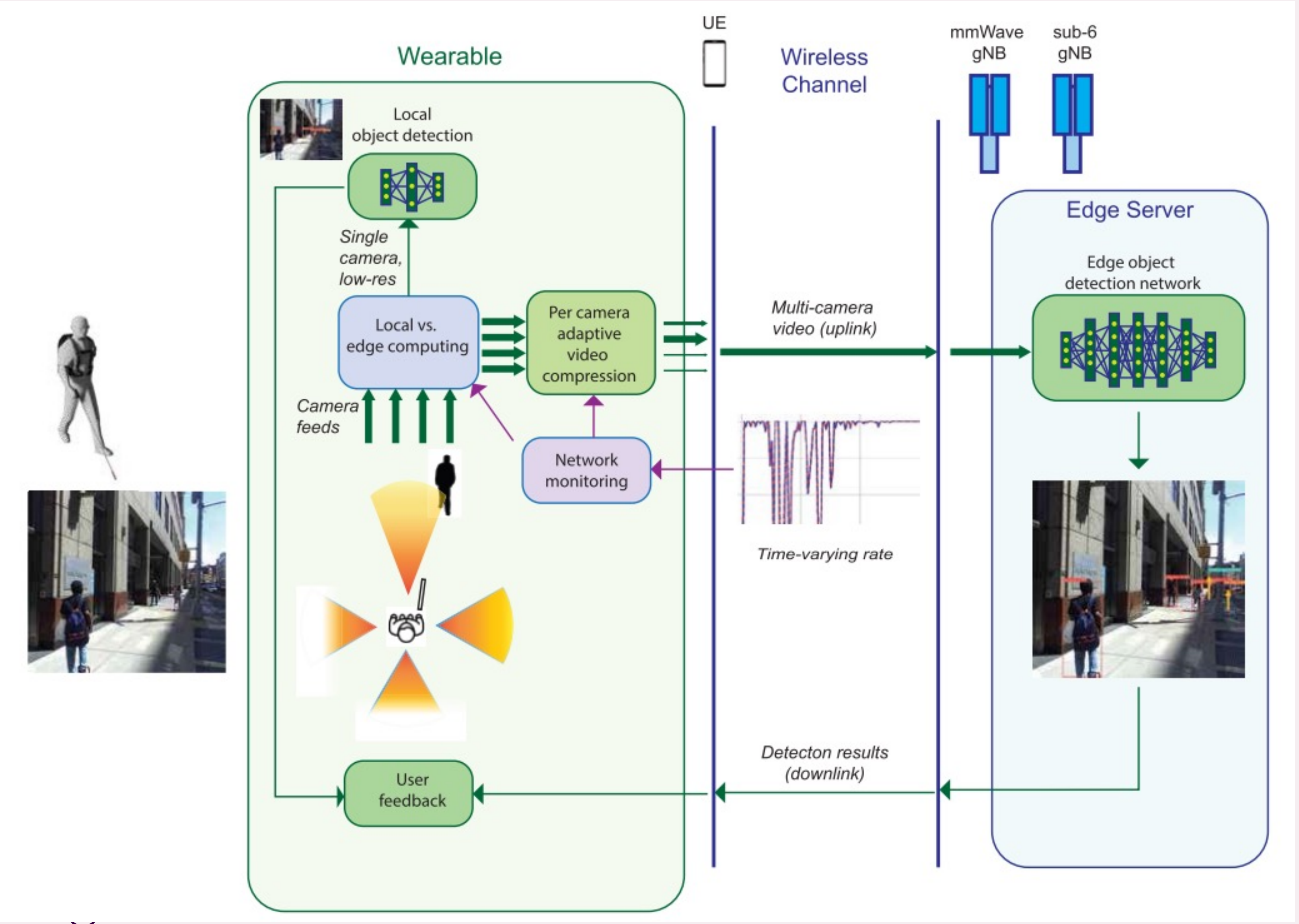
Visual Analytics for Mobile Devices

- Example applications:
  - Object detection and scene understanding to assist people who are blind or with low vision
  - Augmented Reality (AR)
- Challenges:
  - Deep learning models for visual analytics require high performing CPUs and GPUs
  - Mobile devices have limited computing speed and battery energy, may not have GPUs
  - Running the models on mobile devices may not satisfy the real-time processing requirement and can drain the device battery quickly

Solution: Offload Computation to the Edge

- Mobile device transmits image/video to an edge or cloud server
- Server performs analytics tasks and sends the results back to the mobile
- The mobile device may execute the models locally when the network connection is poor
- Compression rate vs. local computing depends on the connection speed and battery status

A wearable system for providing real-time navigation assistance for blind and low vision people

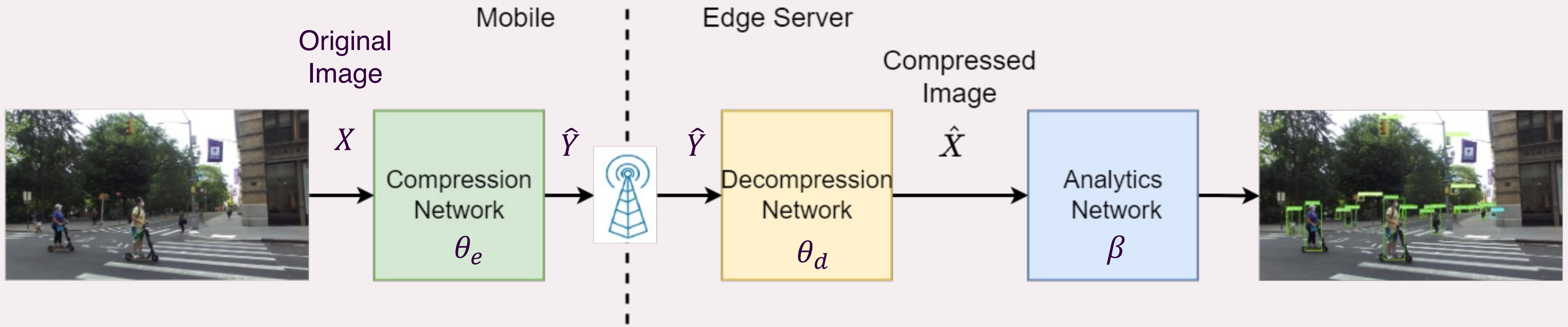


- Z. Yuan, T. Azzino, Y. Hao, Yu, Y. Lyu, H. Pei, A. Boldini, M. Mezzavilla, M. Beheshti, M. Porfiri, Maurizio, T. Hudson, W. Seiple, Y. Fang, S. Rangan, Y. Wang, JR Rizzo, "Network-Aware 5G Edge Computing for Object Detection: Augmenting Wearables to 'See' More, Farther and Faster," in *IEEE Access*, vol. 10, 2022.
- Visual analytics can be done via edge server or locally, on single or multiple views, based on available bandwidth and the wearable battery status
- Supported in part by the NSF Smart and Connected Community Program

What to compress and send?

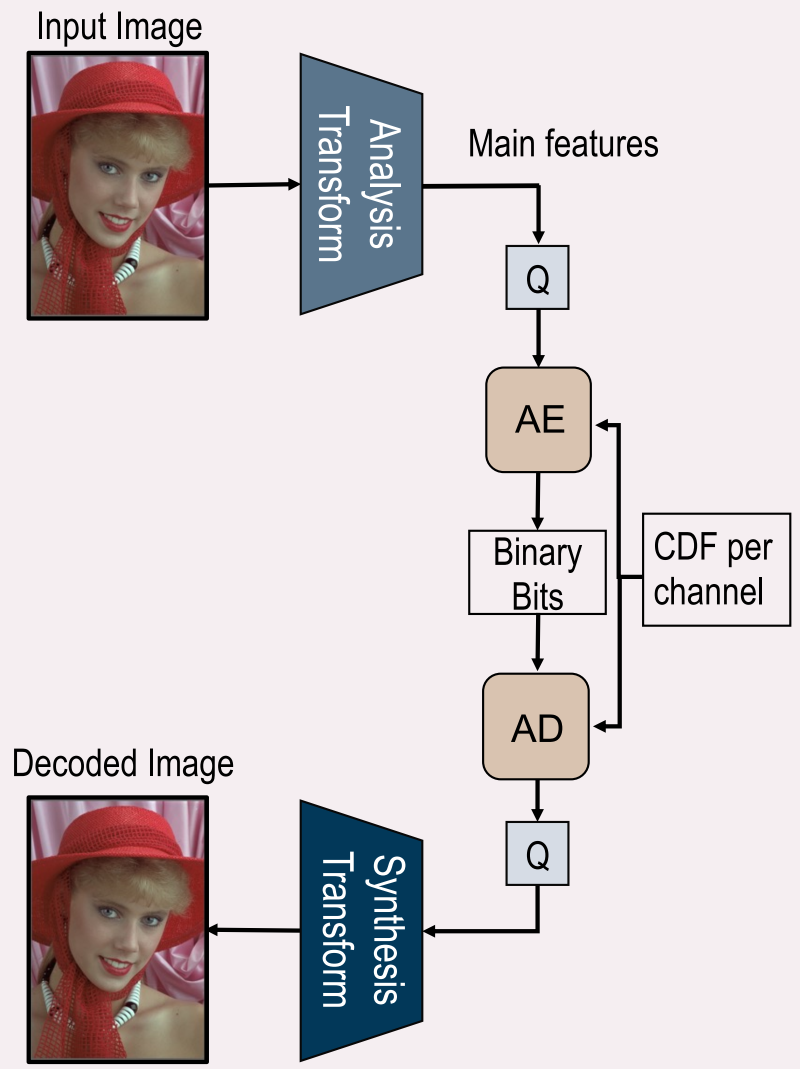
- Two options
  - Mobile compress and send images to the server, server decompress images and run the analytics model
  - Split the task model computation between mobile and server, and compress intermediate features
- Compression rate needs to be adapted to network throughput /mobile battery status
- Related work: MPEG video coding for machine, feature compression

Compression-Decompression-Analytics Framework



- Option for compression
  - Existing codec (e.g. JPEG, BPG): can only refine the analytics task model to handle compression artifacts
  - Learnt codec: can jointly train the codec (θ) and task model (β)

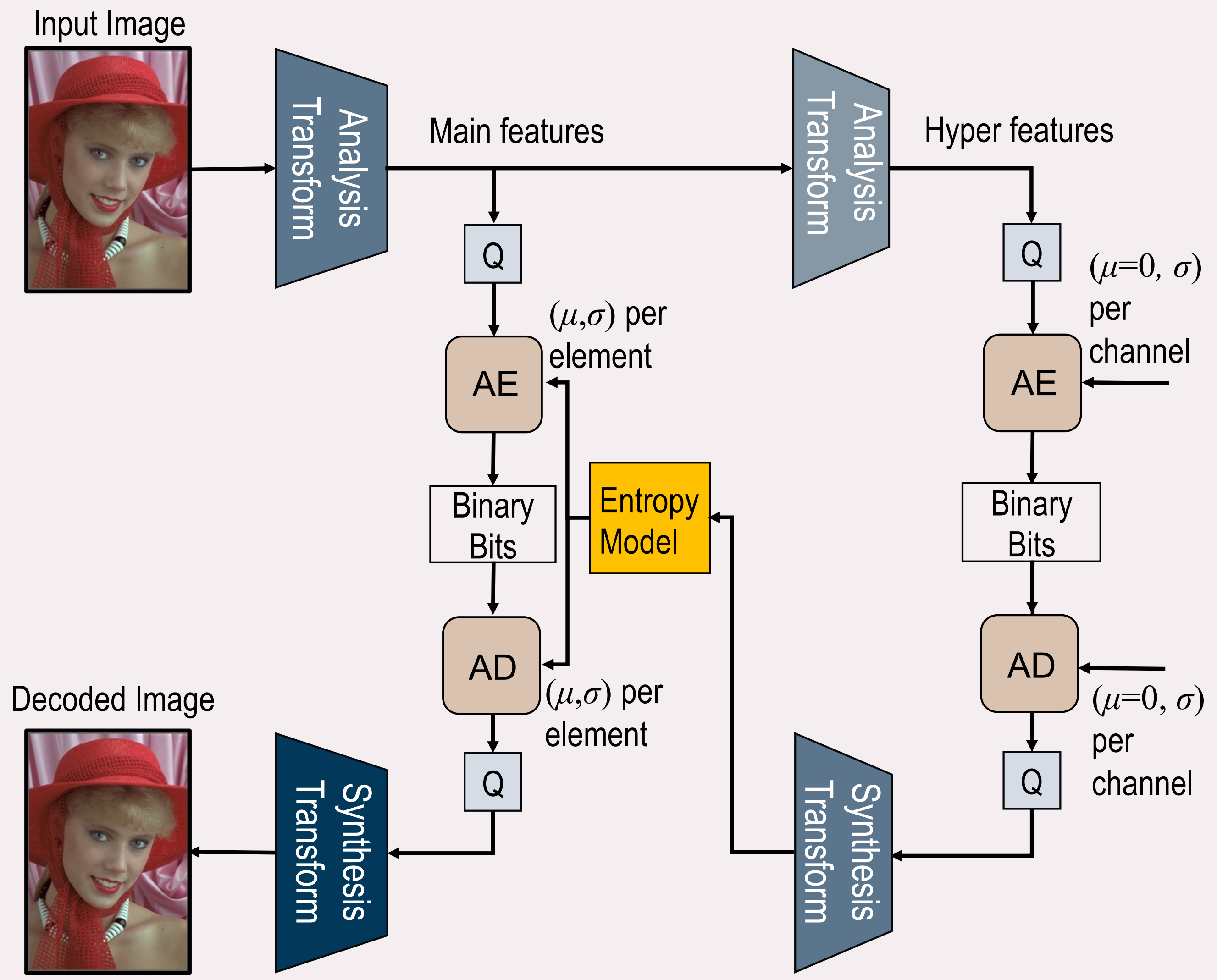
Learnt Compression: Basic Framework of Ballé



- Rate-distortion loss $L = D + \lambda R$
  - $R = -\sum_i \log(\Pr(\hat{F}_i))$
  - $\Pr(\hat{F}_i) = \mathrm{CDF}\left(F_i + \frac{\Delta}{2}\right) - \mathrm{CDF}\left(F_i - \frac{\Delta}{2}\right)$
- Rate is approximated by the cross entropy between the actual probability distribution and estimated distribution.
- Accurate probability modeling is important so that the rate can be accurately estimated and minimized.
- Assume all features in the same feature channel follow i.i.d. distribution, learnt during training.
- CDF described by a piecewise linear function [Balle2016] or a learnt MLP [Balle2018]

J Ballé, V Laparra, EP Simoncelli , End-to-end Optimized Image Compression, ICLR 2017 (arXiv 2016)

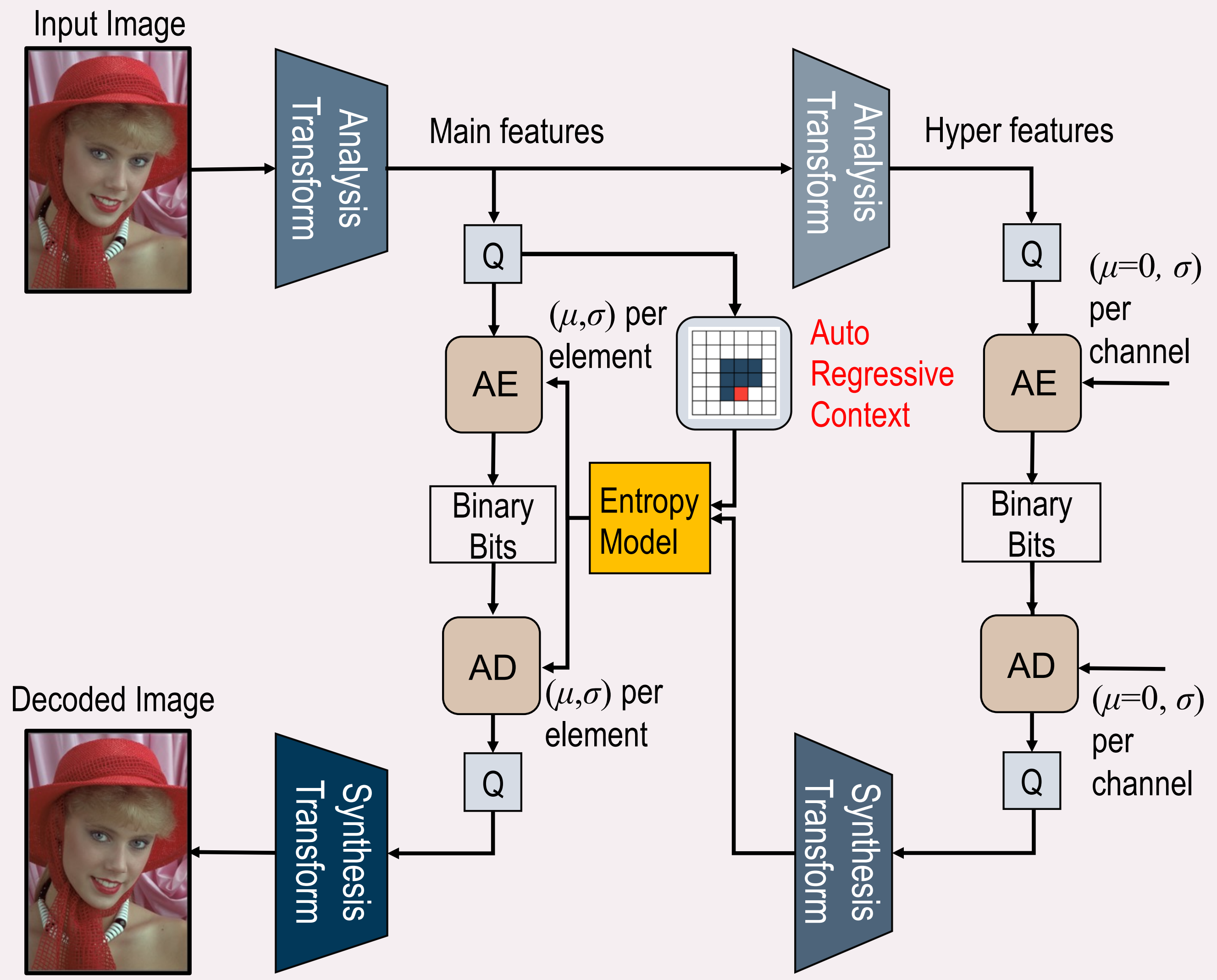
Adding Hyperprior



- Hyperpriors are introduced for more accurate probability modeling for main features
- Assume each main feature element is Gaussian
- Entropy model: Use hyperprior features (side info) to predict the mean and STD of each main feature element

J. Ballé, D Minnen, S Singh, SJ Hwang, N Johnston , Variational image compression with a scale hyperprior, ICLR2018

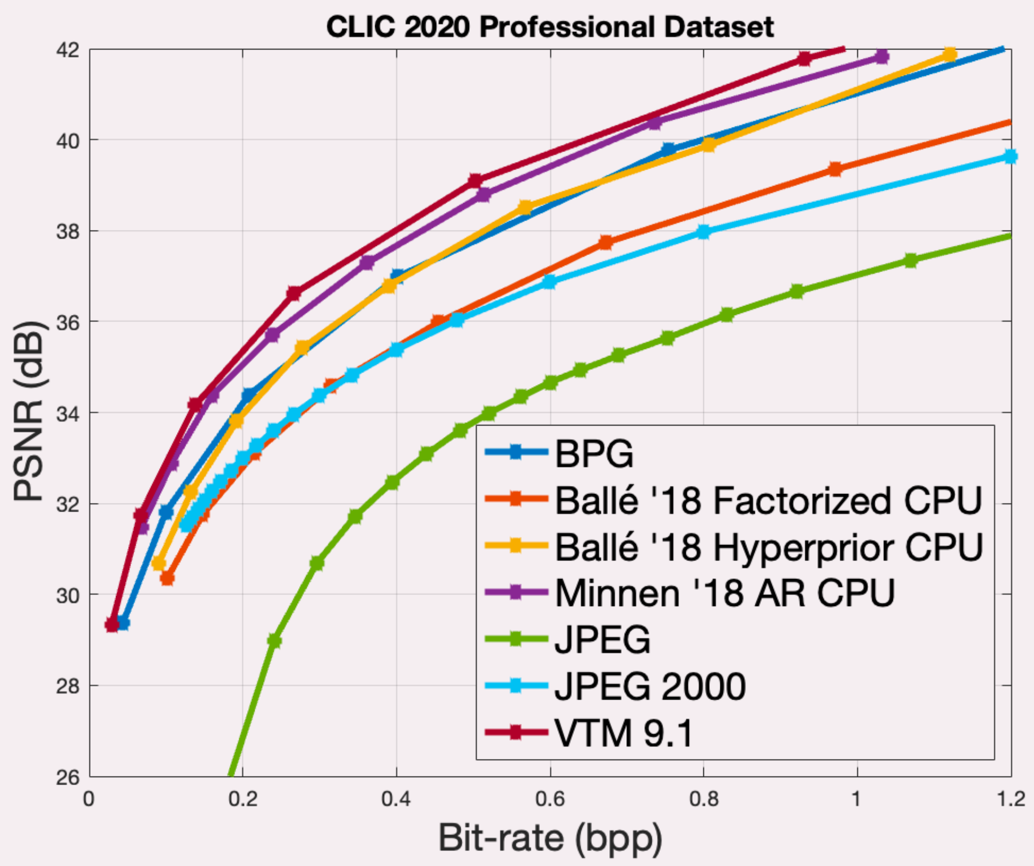
Adding Autoregressive Context

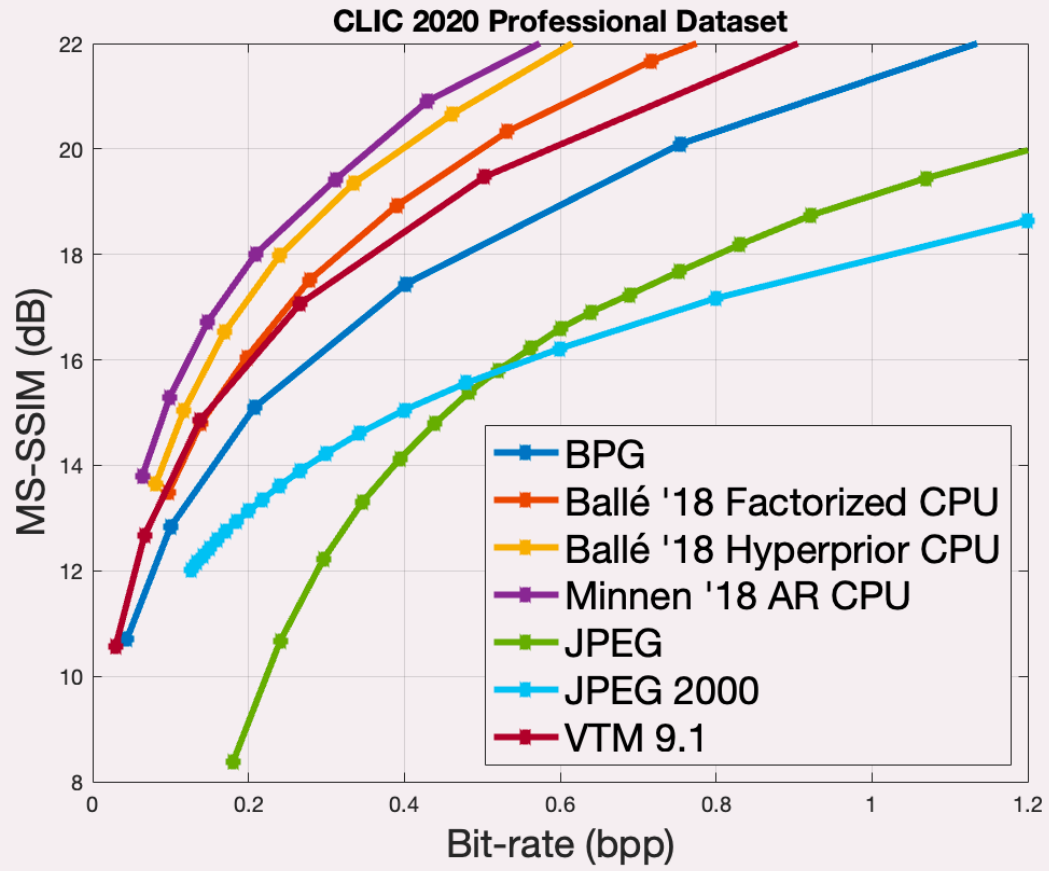


- Akin to Intra-prediction
- Entropy model use both hyperprior features and AR context to predict the mean and STD of each main feature
- Recursive generation of spatial context from decoded features prevents parallel processing using GPU – decoding very slow
- Channel-wise (ChARM) vs. Spatial recursive

D Minnen, J Ballé, GD Toderici , Joint autoregressive and hierarchical priors for learned image compression, NeurIPS 2018

R-D Performance



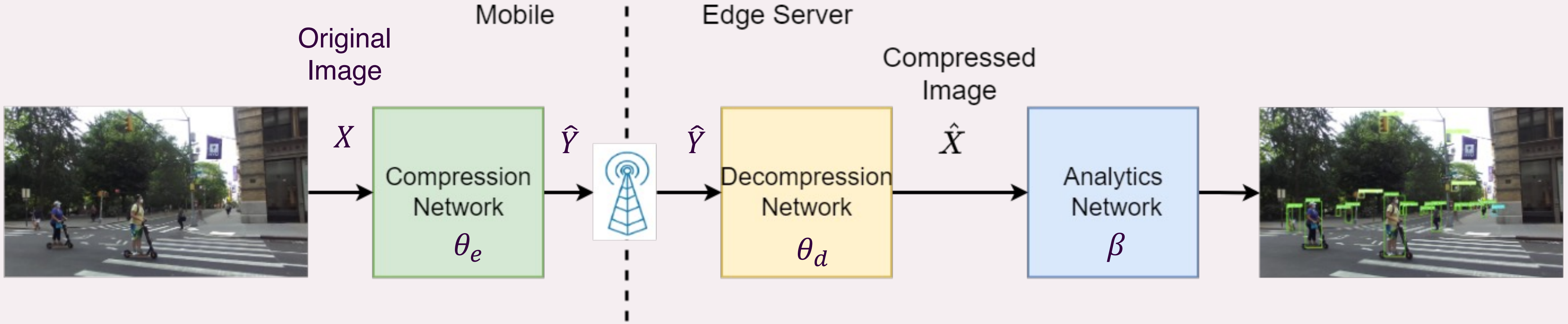


R-D results provided by InterDigital at https://github.com/InterDigitalInc/CompressAI/tree/master/results
Figure prepared by Yueyu Hu.
Learnt coders optimized for perceptual quality metrics can do much better for those metrics!

Why using learning for compression?

- Can optimize all components together for a given rate-quality objective
  - Can learn the optimal nonlinear transform for a target bit rate
- For video: Can learn how to best exploit the temporal redundancy
  - Explicit motion compensation to generate residual
  - Code the current frame directly conditioned on prior frames
- Can be trained to optimize any differentiable perceptual quality metric or using adversarial loss (making decoded images have similar distribution as true images)
- Can also be trained to optimize a visual analytics task under rate constraints!

Compression-Decompression-Analytics Framework



- Option for compression
  - Existing codec: can only refine the task model to handle compression artifacts
  - Learnt codec: can jointly train the codec (θ) and task model (β):
    - $L(x;\theta_e,\theta_d,\beta) = L_{task}(\hat{x}(\hat{y};\theta_d);\beta) + \lambda R(\hat{y}(x;\theta_e))$

      (Task-aware image compression)

Rate-Accuracy Performance

Object detection using YOLO

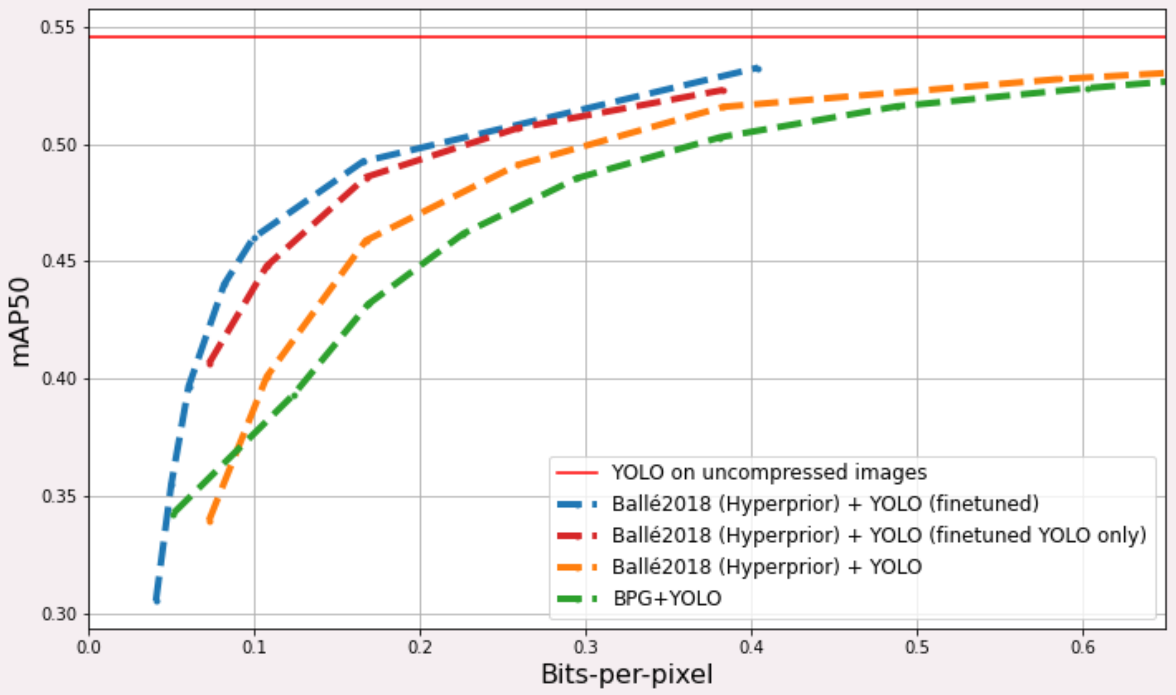
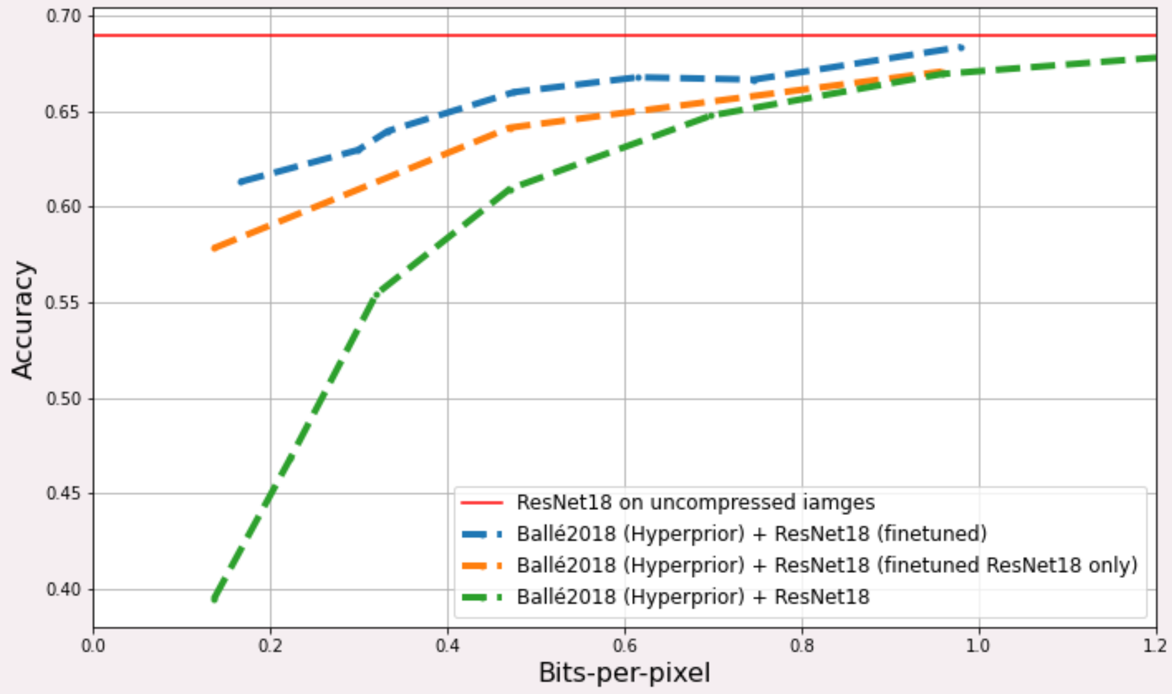
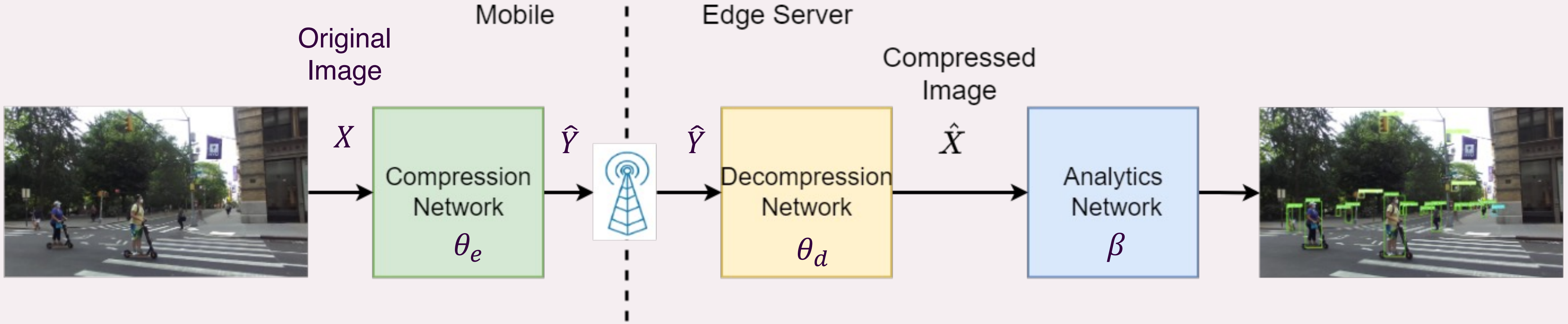


Image classification using ResNet18



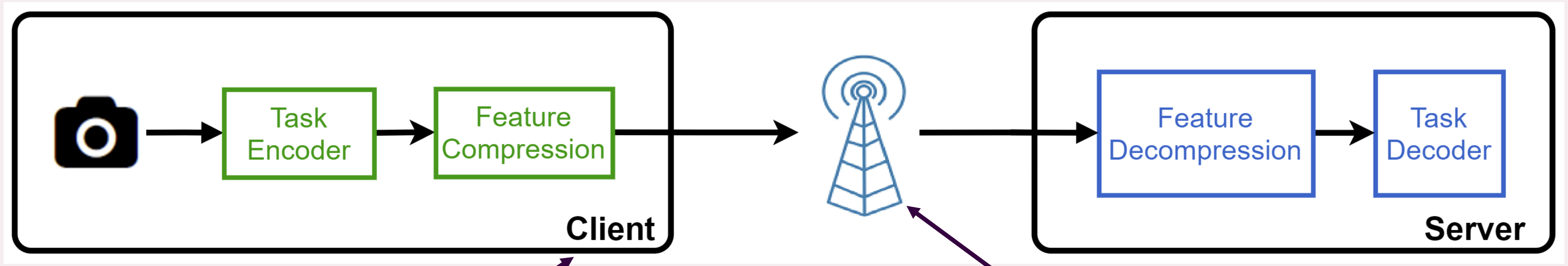
Jointly refine the compression model and the analytics model can significantly improve analytics performance at lower rates!

Problem with the compression-decompression-analytics pipeline



- Compression network is itself complex!
  - Also hard to remove information not relevant for the task
- Decompressing back to image is not necessary!
  - Analytics network will take decompressed images and generate features
  - Why not generate intermediate features and compress them?

Split Computing

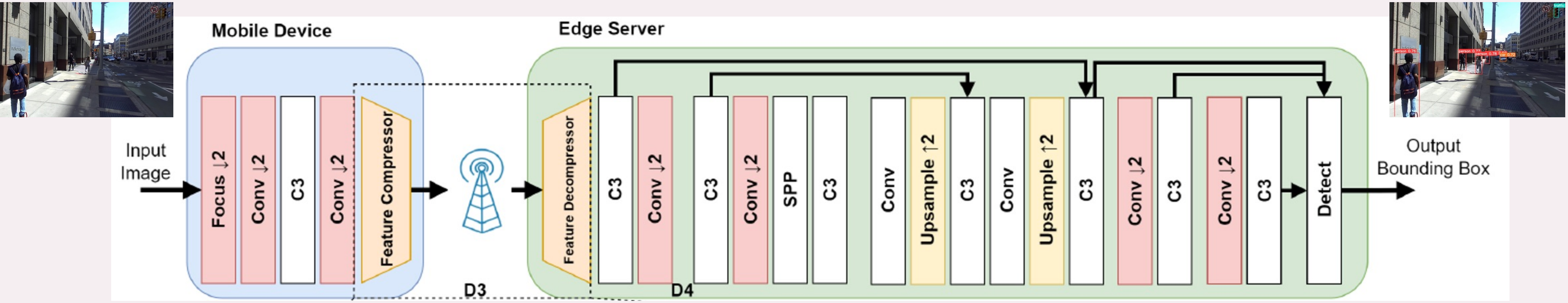


Computationally constrained

Variable b/w wireless link

- Split the visual analytics model into client (task encoder) and server (task decoder) models
- Compress and transmit intermediate features
- Task encoder and feature compressor should have low complexity
- Compression rate should be adaptable to link throughput

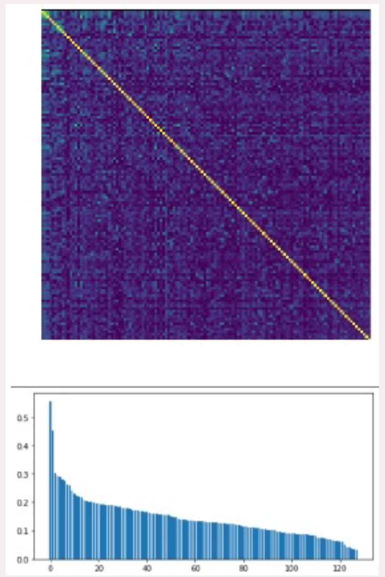
Split Computing for Object Detection



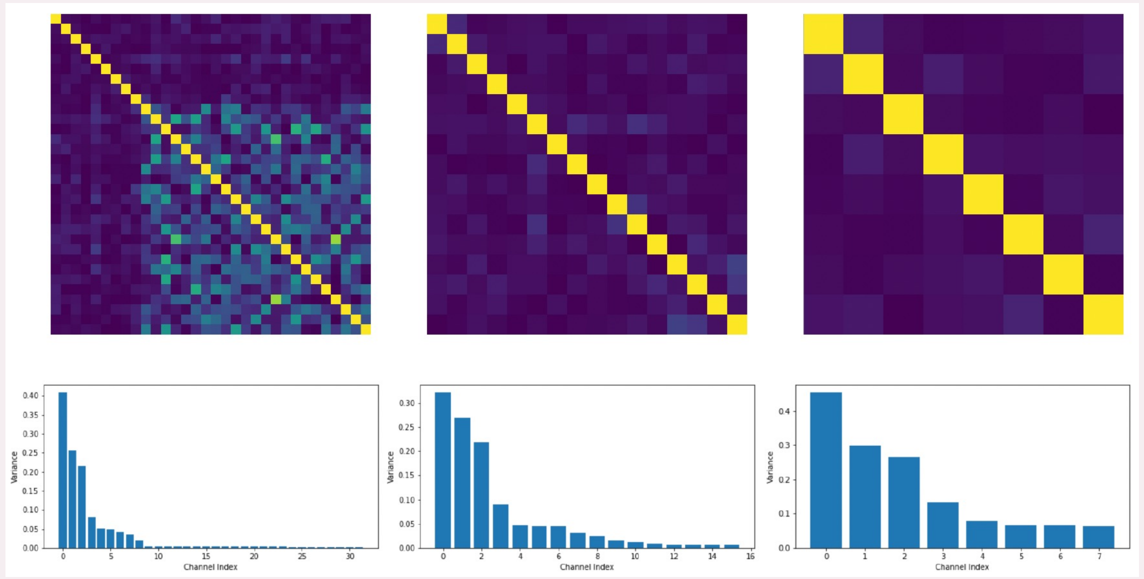
- Use the YOLOv5 model for object detection
- Mobile execute several feature extraction layers of YOLO (task encoder)
  - The mobile should be assigned only a few layers to reduce its computation load (e.g. D3 or D4)
- Mobile compress and send compressed intermediate features to the edge
- Edge server decompress intermediate features and complete remaining YOLO layers (task decoder)

How to compress intermediate features?

- Original features are generated for object detection, not for compression!
- Significant correlation across channels, all channels with high variance.
- Learning transform across channels to reduce correlation and variance
  - End-to-end learnt task-aware PCA
  - Only a small number of channels are necessary to maintain task performance!



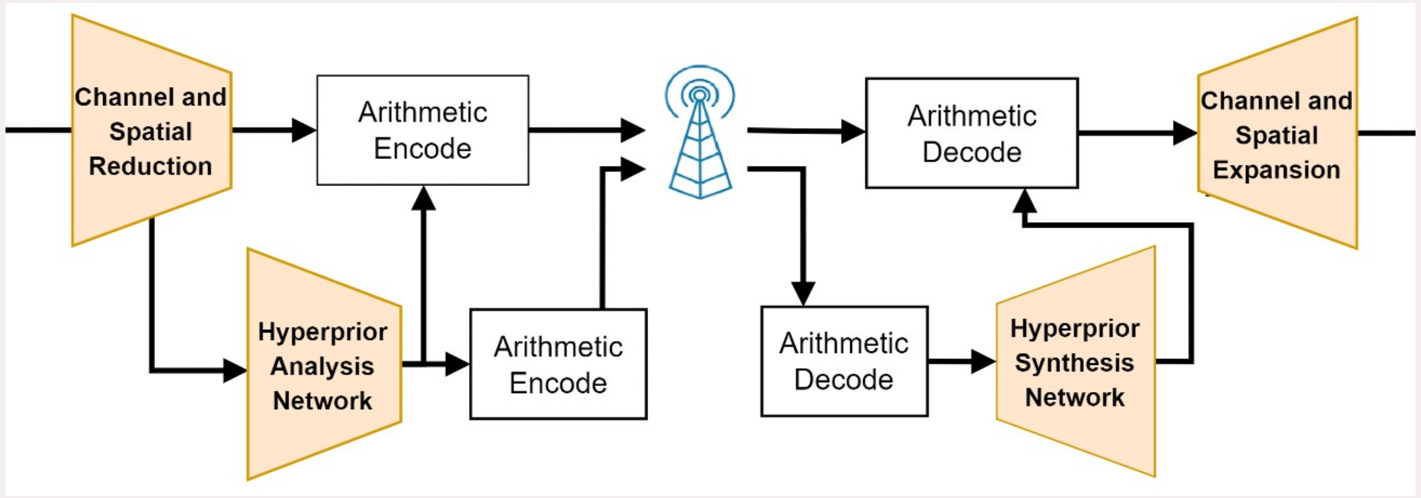
Original 128 channels



32 channels 16 channels 8 channels

- There are also spatial correlations in large feature maps
- Reduce spatial correlation and dimension by strided convolution

Feature Compression


Channel and Spatial Reduction
Arithmetic Encode
Arithmetic Decode
Channel and Spatial Expansion
Hyperprior Analysis Network
Arithmetic Encode
Arithmetic Decode
Hyperprior Synthesis Network

- Compression by reducing spatial and channel dimensions of intermediate feature maps
- Use hyperprior features for probability modeling [Balle2018]

Train the entire model (YOLO+ compression) end-to-end with rate-detection loss

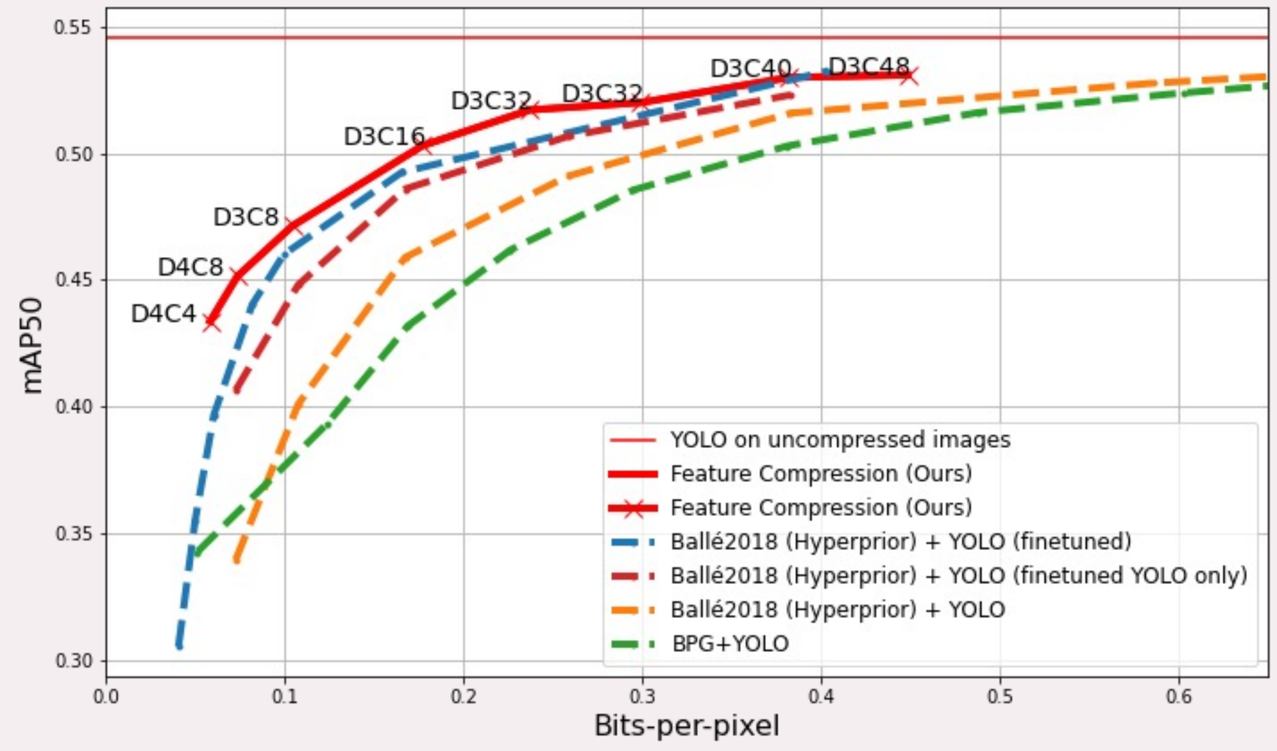
Quantized Reduced feature

$$L = L_R + \lambda \cdot L_{det}$$
$$L_R = \mathbb{E}_{x \sim p_x}[-\log_2 p(\hat{z}(y(x;\theta);\phi))]$$
$$L_{det} = L_{obj} + L_{class} + L_{box}.$$

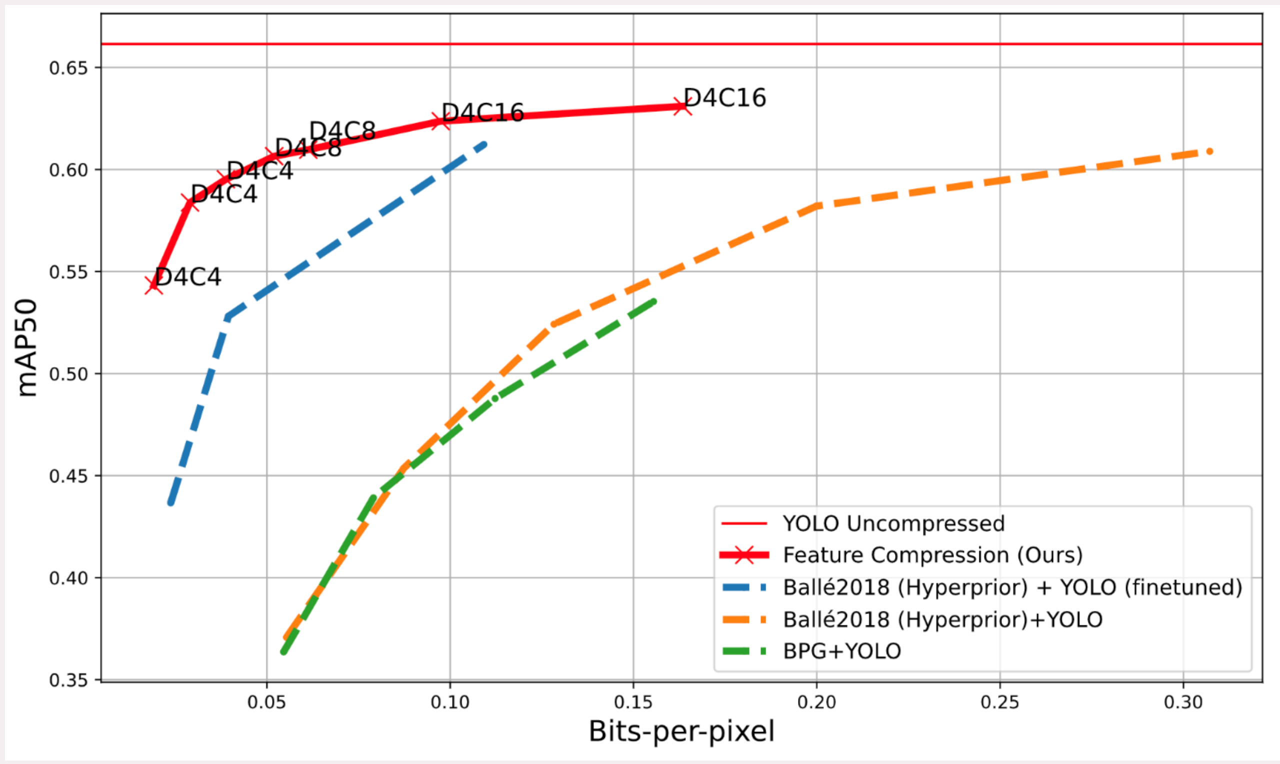
Original Intermediate Features

Image

Perform a search over compression location {D3, D4}, channel numbers N_r, and hyperparameter λ



Full COCO Dataset (80 classes)

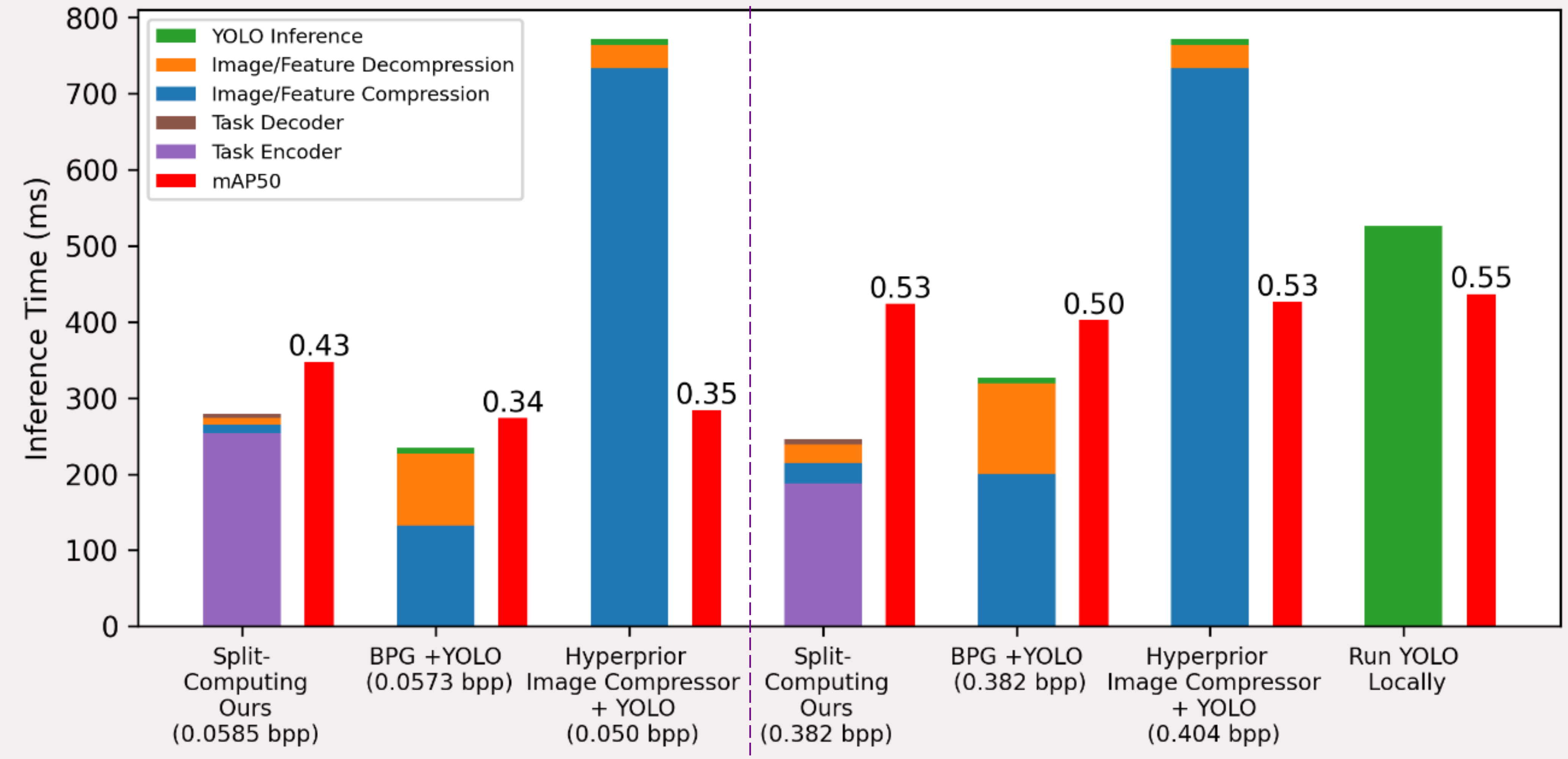


COCO-Traffic Dataset (9 classes)

- Feature compression is better than image compression followed by detection on decompressed images, even when compression is optimized for object detection.
- The number of feature channels can be significantly reduced while retaining acceptable detection accuracy

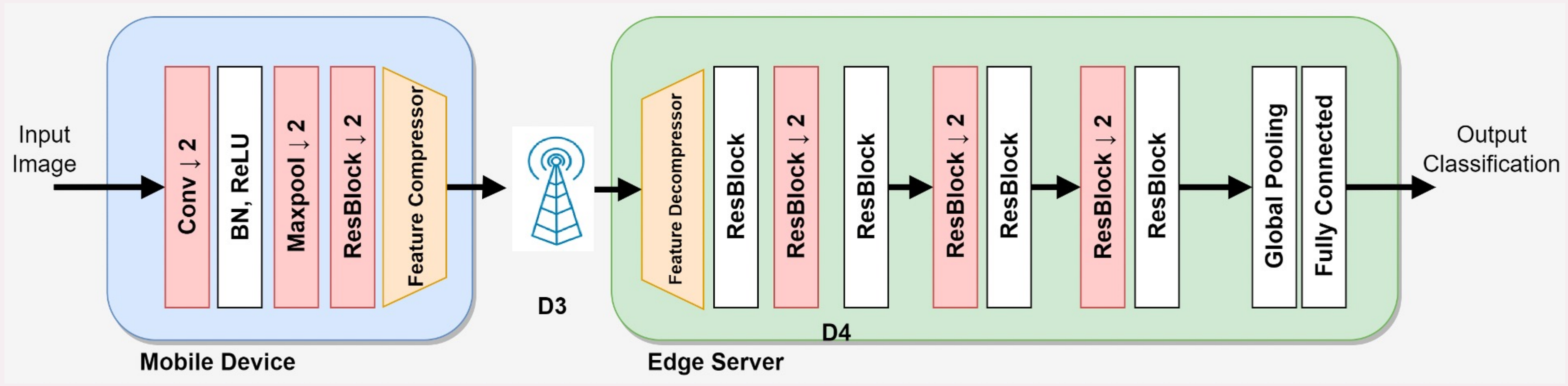
- For applications interested in detecting a small number of object types, feature compression is significantly better (can choose to only retain a few relevant features for selected objects)
- Slower accuracy reduction as the rate drops

Runtime Comparison



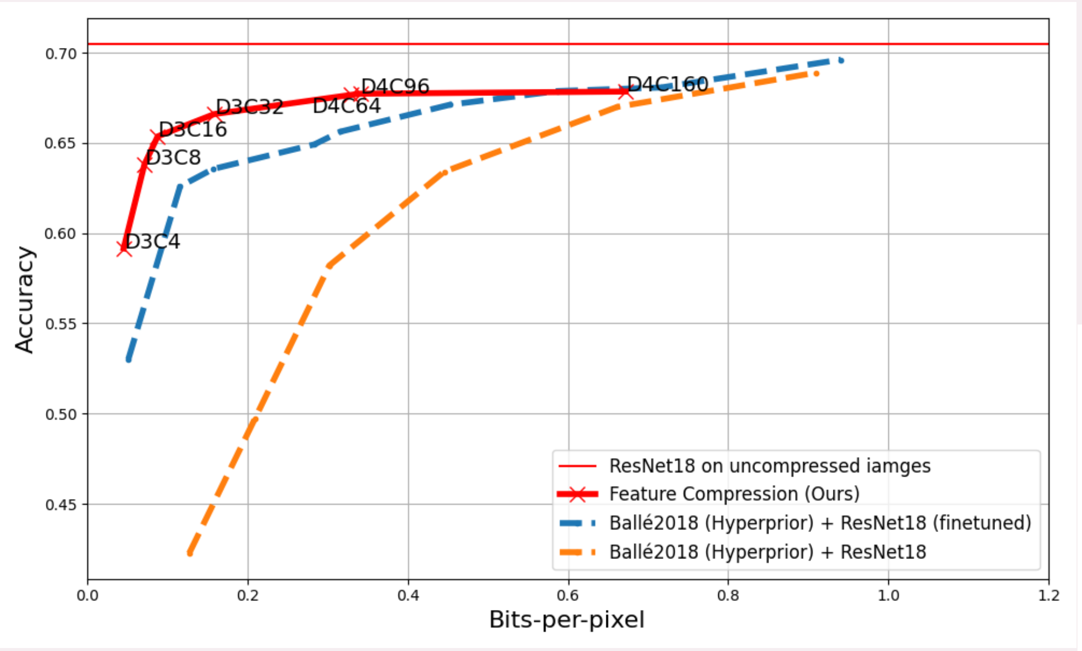
- Measured using a 1.1 GHz CPU for mobile-side computation and an Nvidia RTX 8000 GPU for server-side computation. Runtime in millisecond per image (640x640).
- Compared to running YOLO on the mobile, the mobile and total computation time reduced significantly!
- Learnt image compression + YOLO baseline takes more time than running YOLO locally!
- BPG+YOLO takes about same time, but has lower detection performance.

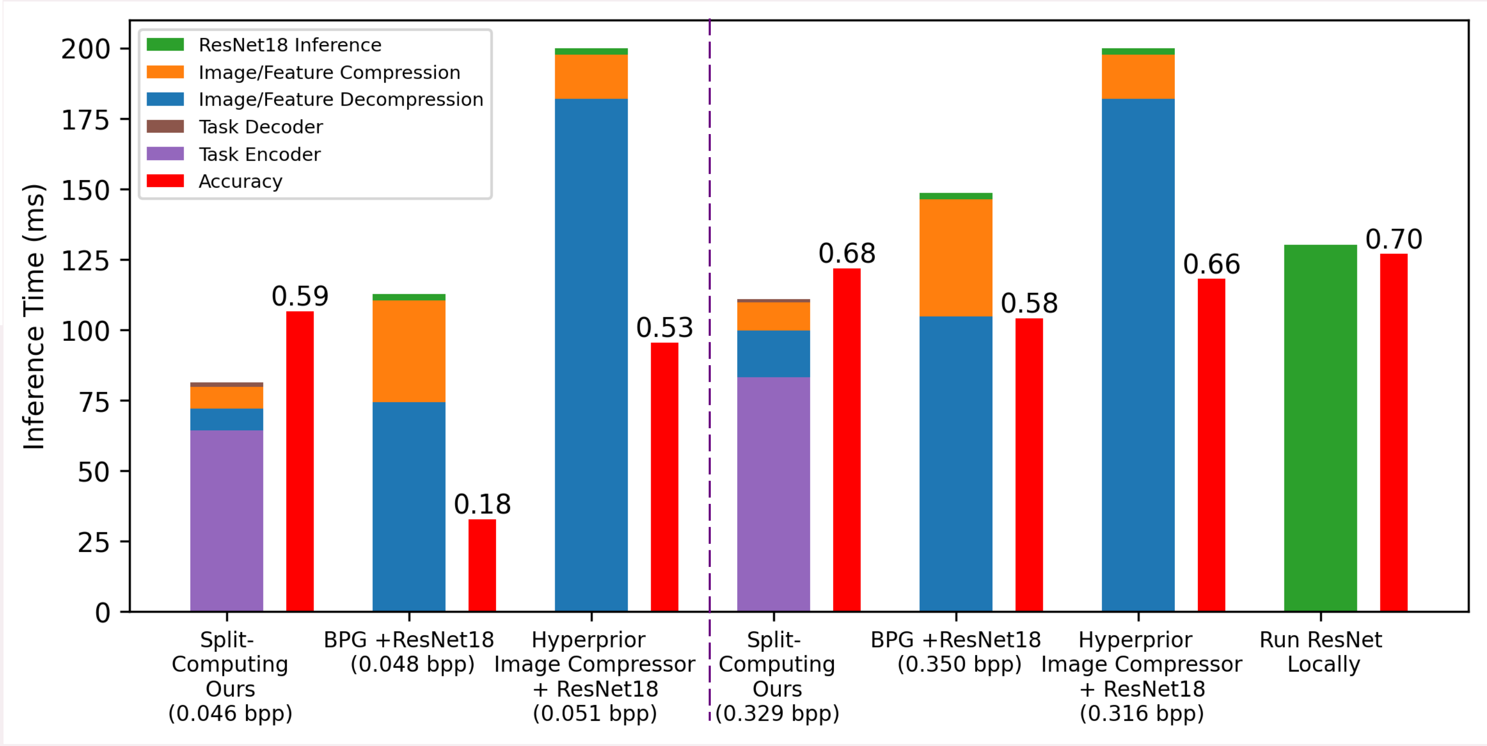
Split computing for image classification



- Using ResNet18 for classification (deeper models may be used in practice to benefit from powerful servers at the edge

Rate-Accuracy and Runtime





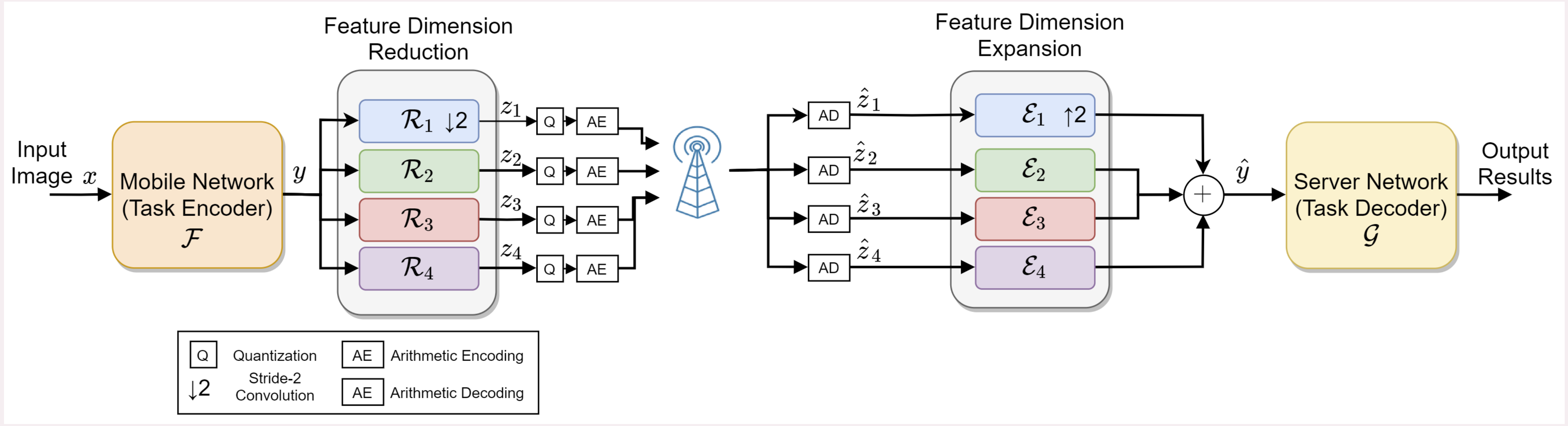
Problems?

- ☐ Previous results are obtained by training a different set of models (task model and feature compression model) for each bit rate
- ☐ Saving multiple models requires significant storage at the mobile
- ☐ Switching between different models may be too slow to adapt to fast varying network environments
- ☐ In multi-cast scenario with different servers/receivers, multiple independent compressed feature streams need to be generated and sent

Scalable Feature Compression

- Generate an ordered set of feature channels:
  - base layer contains first few channels, each enhancement layer contains next few channels
  - Each additional layer improves the analytics performance
- Reduce memory cost and switching time between non-scalable models
- Adapt the number of layers based on the transmission throughput
- Facilitate multi-cast: mobile send all layers once, receivers with different bandwidths can receive different number of layers.

Proposed Approach

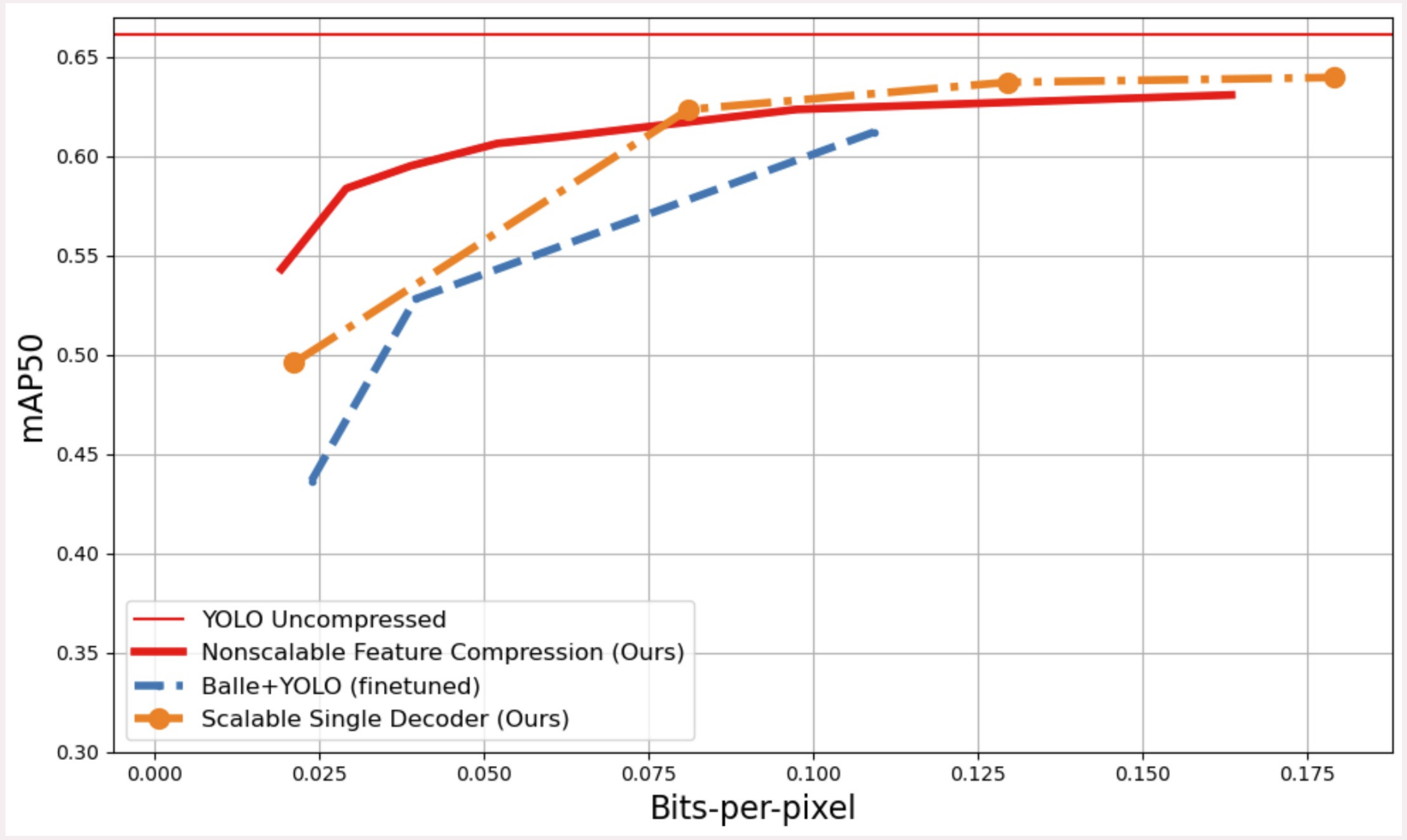


- Generate each layer through a channel/spatial reduction layer ($\mathcal{R}_i$)
- Features in each layer are separately entropy coded using their own hyperpriors
- Received feature layers at the server are expanded ($\mathcal{E}_i$) and added together to form input for the task decoder
- Task encoder and decoder remain the same regardless how many layers are received.

How to train the model to generate scalable layers?

- One idea (borrowing from layered image compression):
  - First train the task encoder and decoder and the feature reduction and expansion modules for the base layer to optimize the rate-task performance with the base layer only
  - Train the feature reduction and expansion modules for the next layer successfully to optimize task performance improvement for additional rate
- Problem: The task encoder and decoder are optimized only for the base layer
- Solution: Train the task encoder, decoder and all feature reduction and expansion modules to optimize for average rate-task performance with different numbers of layers

Rate-Accuracy Results for Object Detection

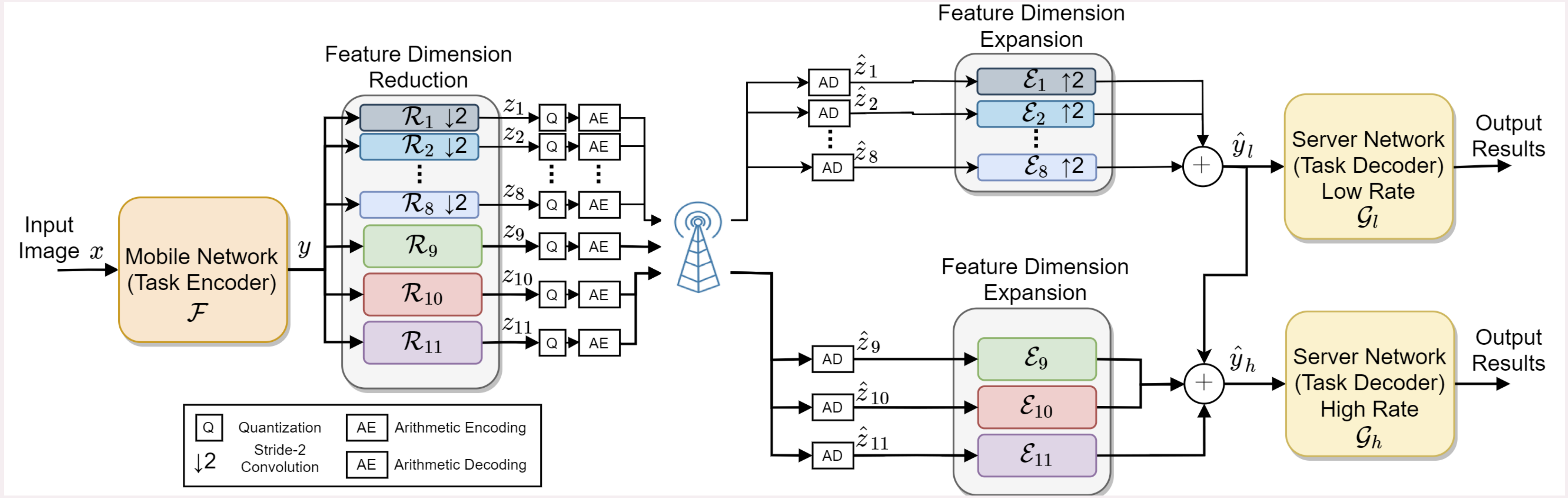


- Trained the model on the subset of COCO data with traffic-related classes
- Scalable model matched the performance of a set of single-rate models in high rate range
- But at low bitrates, the scalable model underperformed significantly

Improving the performance over the entire rate range

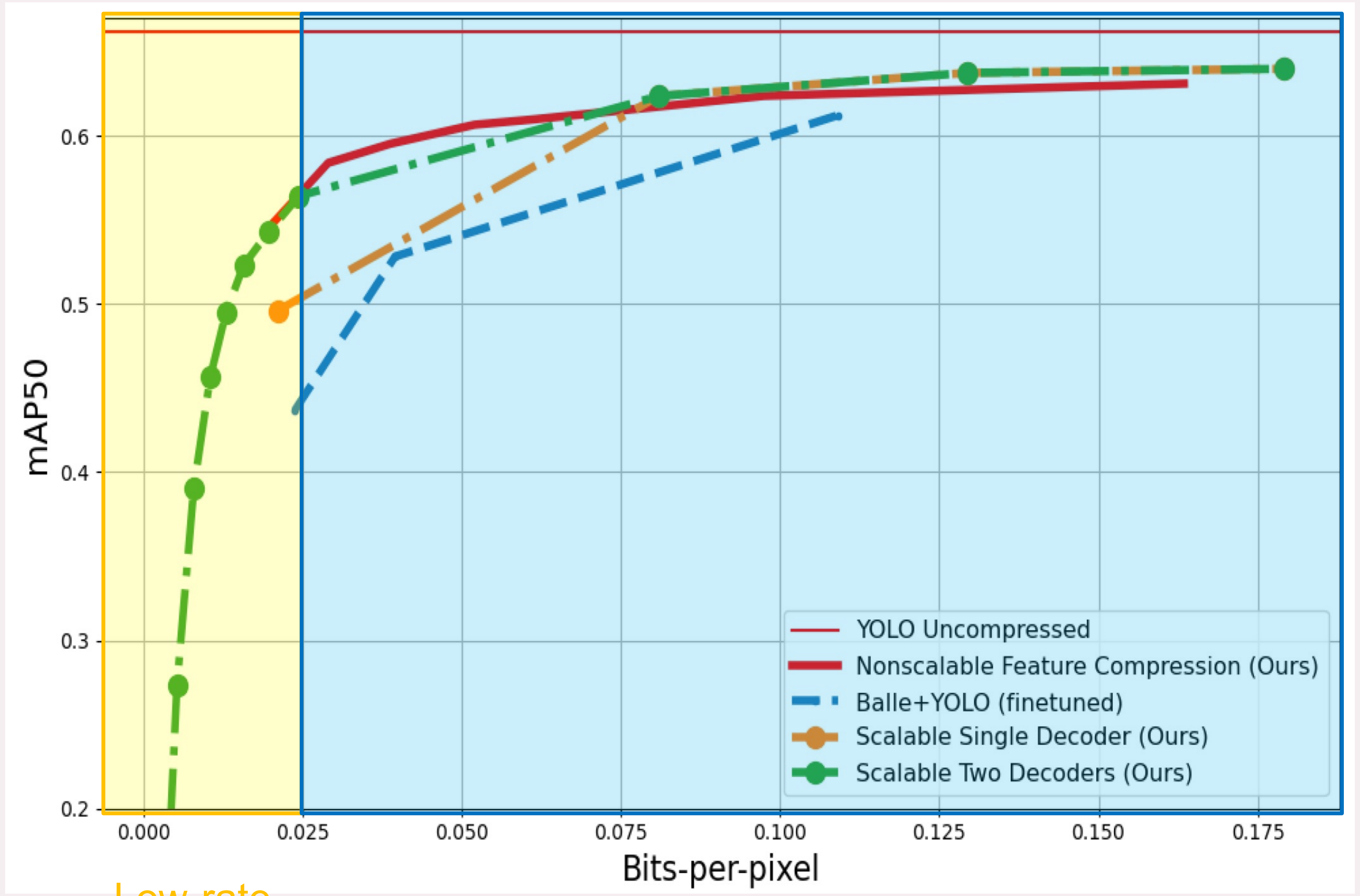
- It is difficult for the same task encoder and task decoder to operate with high performance over a large rate range
- Since computation resource is abundant at the server, we can train multiple task decoders for different rate ranges
- A single task encoder is necessary for generating layered bit streams

Using Two Task Decoders


Input Image x
Mobile Network (Task Encoder) $\mathcal{F}$
y
Feature Dimension Reduction
$\mathcal{R}_1 \downarrow 2$
$\mathcal{R}_2 \downarrow 2$
$\mathcal{R}_8 \downarrow 2$
$\mathcal{R}_9$
$\mathcal{R}_{10}$
$\mathcal{R}_{11}$
z_1
z_2
z_8
z_9
z_{10}
z_{11}
Q
AE
AD
$\hat{z}_1$
$\hat{z}_2$
$\hat{z}_8$
$\hat{z}_9$
$\hat{z}_{10}$
$\hat{z}_{11}$
Feature Dimension Expansion
$\mathcal{E}_1 \uparrow 2$
$\mathcal{E}_2 \uparrow 2$
$\mathcal{E}_8 \uparrow 2$
$\mathcal{E}_9$
$\mathcal{E}_{10}$
$\mathcal{E}_{11}$
$\hat{y}_l$
$\hat{y}_h$
Server Network (Task Decoder) Low Rate $\mathcal{G}_l$
Server Network (Task Decoder) High Rate $\mathcal{G}_h$
Output Results
Q Quantization
AE Arithmetic Encoding
$\downarrow 2$ Stride-2 Convolution
AE Arithmetic Decoding

- Split the original base layer into 8 small layers (1 channel each, with down sampling)
- Using a low-rate task decoder for up to 8 layers
- Using a high-rate task decoder (previously trained) for more than 8 layers

Results with Two Task Decoders



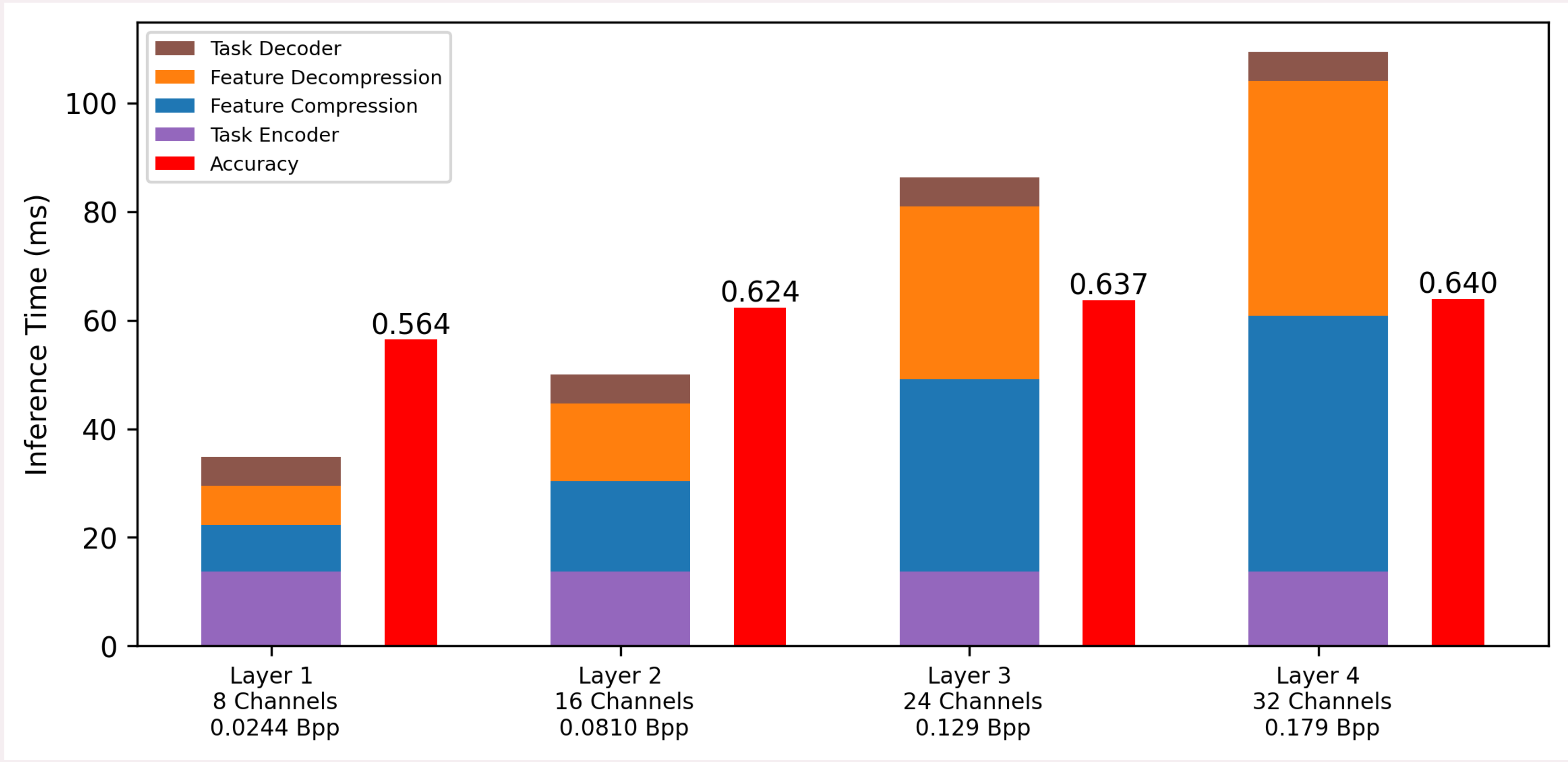
Low-rate decoder

High-rate decoder

- The model achieved comparable performance compared to non-scalable models throughout the entire bitrate range.
- The same task encoder is used for all points on the curve, while the task decoder is switched depending on the target rate.

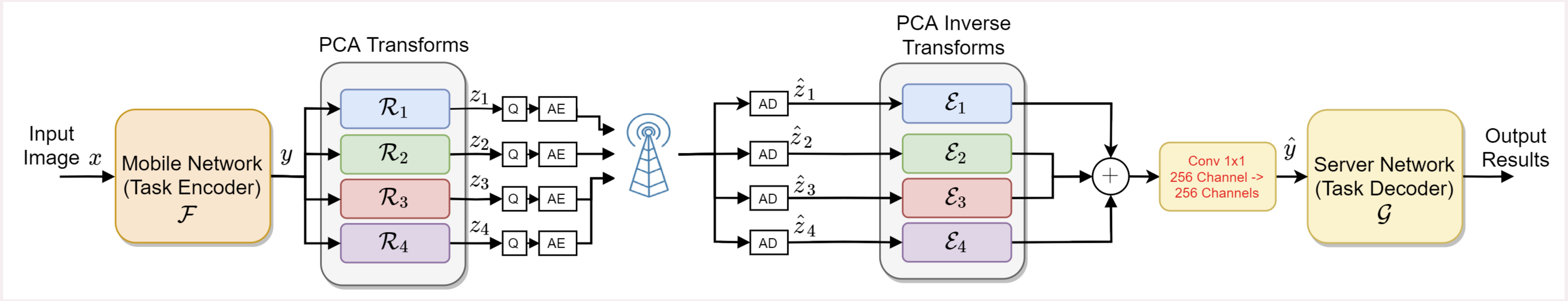
Complexity Scalability

- The proposed model is also scalable in complexity
- Each additional layer requires more computation and more battery energy consumption
- The number of layers can be tuned based on the mobile battery status in addition to transmission throughput

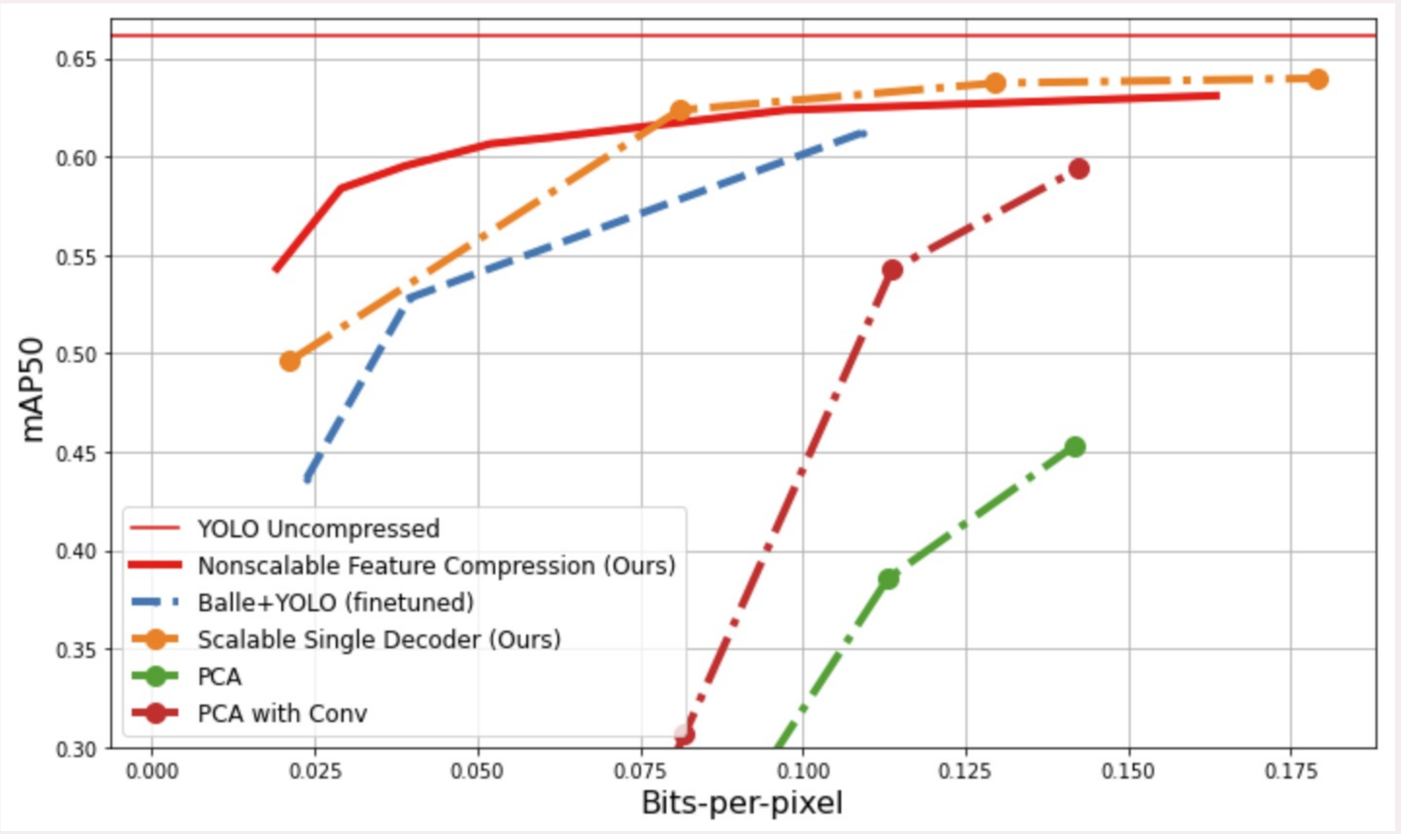


Can we use PCA to generate layered features?

- Instead of training the feature reduction and expansion layers, can we use PCA on y to generate uncorrelated and scalable layers z_l ?
  - Given the number of channels, PCA minimizes MSE between $\hat{y}$ and y
- Since PCA reconstructs y to optimize MSE, an additional 1x1 conv is introduced to transform y into the expected input for the task decoder to see if it improves the task performance



PCA for Scalable Feature Compression

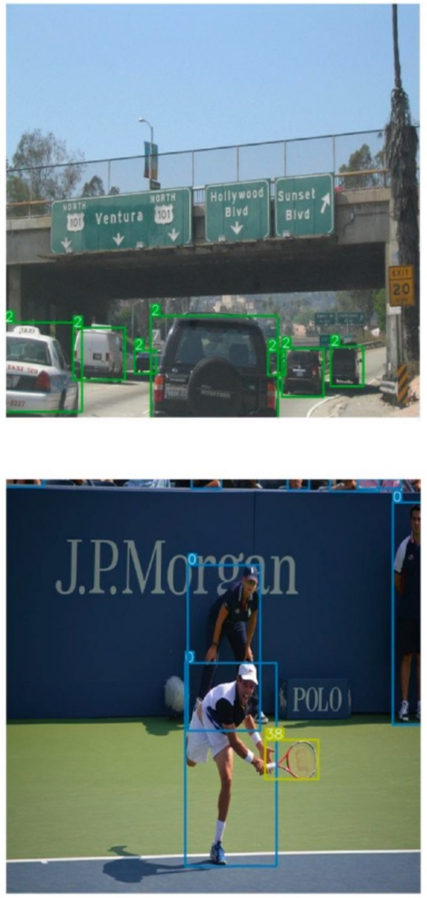


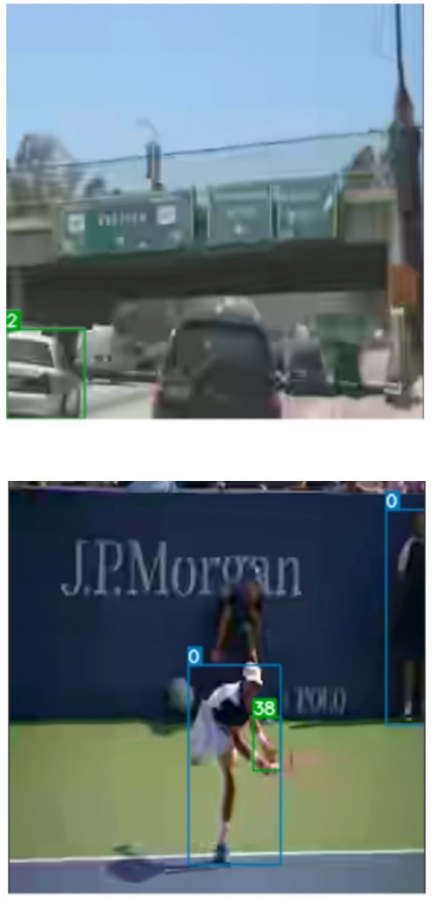
- With the same task encoder and decoder, PCA is significantly worse than the learnt transforms optimized for the task performance!
  - Transform optimized for MSE does not lead to optimal task accuracy
  - Adding one learnt transform after PCA reconstruction provides improvement
- Jointly training task encoder/decoder and feature reduction/expansion layers is necessary for good task performance

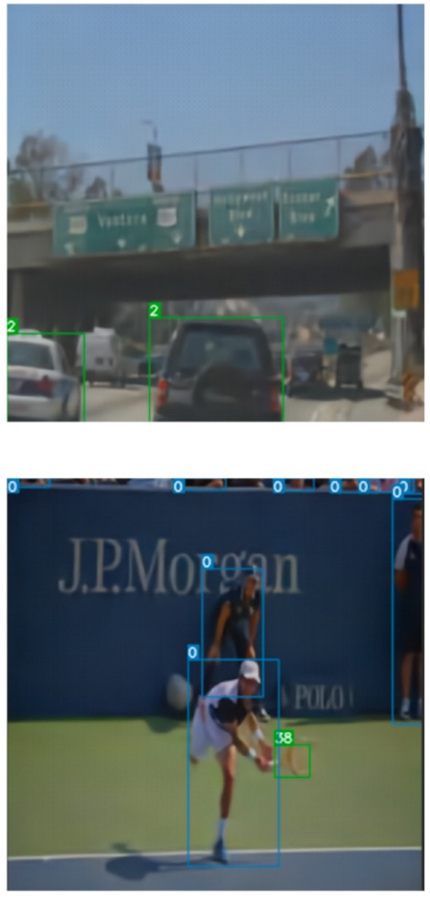
Can we reconstruct images from task features?

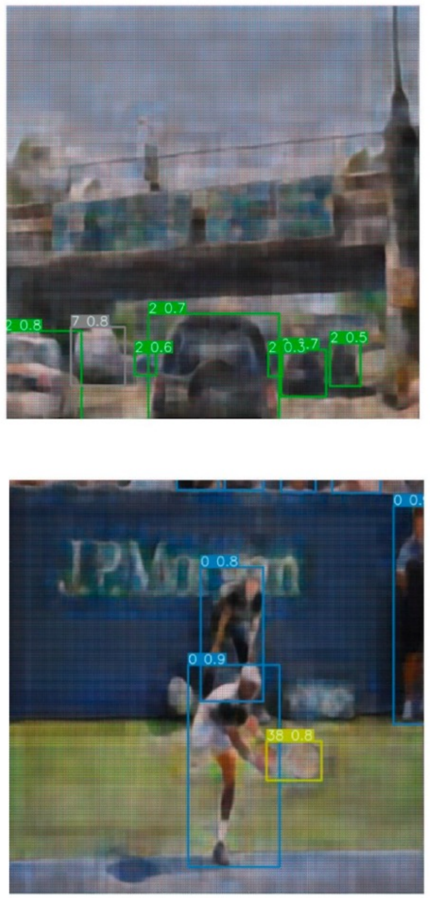
- Help the user at the server to visualize and verify the object detection result
  - Low quality is acceptable
  - Do not want to degrade object detection performance
- Train a convolutional network (reverse of the YOLO backbone) to recover the image from the compressed features
- Help us to understand what information is carried by the compressed features

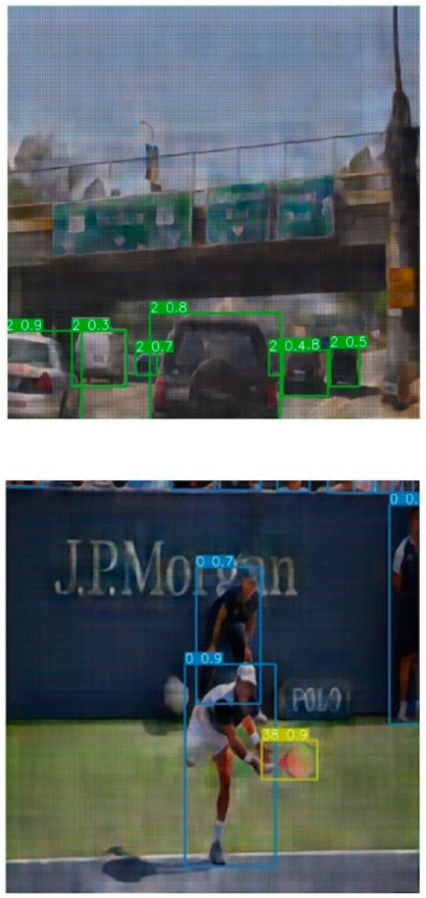
Sample Image Reconstruction Results


Original


BPG, 0.173bpp


Learnt image compression 0.166bpp


Proposed (D3C16) 0.177 bpp


Proposed (D3C48) 0.449 bpp

At low rates, feature compression distorts the background but retains salient object characteristics

Task-aware learnt image compression spends unnecessary bits to retain background information

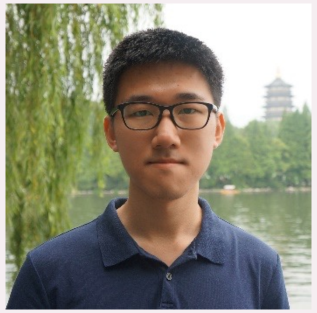
Concluding Remarks

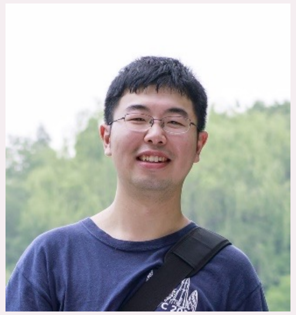
- For edge-assisted visual analytics, split computing appears more promising than compression-decompression-analytics framework
  - Better rate-accuracy trade-off, at substantially lower complexity at the mobile
- Rate-and complexity scalable compression can achieve similar rate-analytics performance as non-scalable compression with multiple models
- Proposed feature compression through channel and spatial reduction is generally applicable for feature compression for different tasks and models
- Open challenges:
  - Can we learn and compress features that are good for all important analytics tasks
    –> perceptual quality models for visual understanding ?

References

- Z. Yuan, S. Rawlekar, S. Garg, E. Erkip and Y. Wang, "Feature Compression for Rate Constrained Object Detection on the Edge," *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2022, pp. 1-6, doi: 10.1109/MIPR54900.2022.00008.
- Z. Yuan, S. Garg, E. Erkip, Y. Wang. "Scalable Feature Compression for Edge-Assisted Object Detection Over Time-Varying Networks". In *MLSys 2023 Workshop on Resource-Constrained Learning in Wireless Networks*.
- Z. Yuan, T. Azzino, Y. Hao, Y. Lyu, H. Pei, A. Boldini, M. Mezzavilla, M. Beheshti, M. Porfiri, T.E. Hudson, W. Seiple, Yi, Fang, S. Rangan, Y. Wang, J.R. Rizzo, "Network-Aware 5G Edge Computing for Object Detection: Augmenting Wearables to "See" More, Farther and Faster," in *IEEE Access*, vol. 10, pp. 29612-29632, 2022, doi: 10.1109/ACCESS.2022.3157876.

Thanks


Jacky Yuan


Yueyu Hu

Siddharth Garg

Elza Erkip

- https://wp.nyu.edu/videolab